AMENDMENTS TO THE SPECIFICATION

Please amend the specification as follows:

In the paragraph on page 3, lines 1-14:

Figure 1 is a data-flow diagram of memory failure within such a IA-64 COTS computer system 100. A typical IA-64 computer system 100 includes a kernel process 102 in communication with a large number of other computer processes, such as process 104, over an input-output (I/O) channel 106. In response to a soft memory error 108 which corrupts process 104, the kernel 102 generates an MCA signal 110 which typically requires that process 104 be terminated. If process 104 served an application or [[other]] some other top-level program, or utility, such programs or utilities will then terminate, perhaps resulting in a substantial loss of important data which had not yet been saved. Even worse, process 104 could have been a key operating system process which causes a system crash, requiring that the whole computer be rebooted. Such a drastic action not only results in a loss of important data and perhaps termination of network communications, but also results in a significant loss of time to the computer's 100 users, who must not only reboot the computer, but also bring up the application programs again and perhaps re-enter data.

In the paragraph on page 3, line 16 through page 4, line 3:

Lock-step processors, mentioned above, are one approach toward implementing [[a]] fault-tolerant computing systems which can perhaps recover from some design and hard errors. Lock-step processors are found within Compaq Himalayas Non-Stop Series of computers and IBM's S/390 computer series. Lock-step processor systems include two hardware processors strictly synchronized cycle-by-cycle. They execute exactly the same instruction each cycle. Lock-step systems also include a substantial amount of internal circuitry inside each of the processors for internally checking that the two lock-stepped processors are indeed operating consistently. Lock-step processors, however, are still vulnerable to memory hard and soft errors since the two processors share memory resources. Thus, if the shared memory fails, the lock-step processors will not be able to recover ~~can not recover~~ and the computer must be rebooted. Even further, lock-step processor systems are very expensive, since duplication of very expensive and necessarily complex circuitry is required.

In the paragraph on page 4, line 15 through page 5, line 6:

As a final example, Cornell University has developed a fault-tolerant computing technique based on "Hyper-Visors." A Hyper-Visor is a software virtual machine that is instantiated between a computer's processor and the computer's operating system, and gives the illusion of multiple processors on one processor. In a typical fault-tolerant Hyper-Visor implementation, the processor hosting a copy of the Hyper-Visor, is part of a complete system, but [[gives]] the Hyper-Visor gives the illusion of multiple processors sharing the rest of the system. The Hyper-Visor implements two or more processors, each of which is able to run its own operating system, application program, and utility processes. During normal operation, only the first virtual processor interacts with system software and resources. Upon a failure on the first virtual processor, however, the backup virtual processor takes over and processing continues. Like the fail-over cluster technique, all application jobs are switched over to the other Hyper-Visor processor. However, since virtual processors are sharing resources, such as memory and disks, errors in these may affect both virtual machines. Lastly, virtual machines must present a fault isolation boundary to be effective for fail-over support. Unfortunately, this requires hardware support for the virtual machine monitor and critical system errors such as memory errors may not be isolatable.

In the paragraph on page 6, lines 8-15:

Within the system of the present invention, a primary process memory space hosts a primary process; a duplicate process memory space hosts a duplicate process corresponding to the primary process; a synchronization buffer [[for]] keeps the duplicate process in synchronization with the primary process; a processor generates an exception signal in response to detection of a memory failure condition which affects the primary process; and an operating system receives the exception signal, terminates the affected primary process, and maintains a predetermined number of primary and duplicate processes.

In the paragraph on page 9, lines 18-22:

Also since different fault-tolerance values may be assigned to either parent or child primary processes, some implementations of the present invention may have a parent primary process with only one duplicate, but a corresponding child primary process with three or more

duplicates. Alternatively, the child primary processes can have [[few]] fewer duplicate processes than a corresponding parent primary process.